

**Interactive and Cooperative Visualization of
Unsteady Fluid Flow**

Michael J. Gerald-Yamasaki

RNR Technical Report RNR-92-018, March, 1992

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, California 94035-1000
yamo@nas.nasa.gov

March 27, 1992

Interactive and Cooperative Visualization of Unsteady Fluid Flow

Michael J. Gerald-Yamasaki

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, California 94035-1000
yamo@nas.nasa.gov

March 27, 1992

Abstract

Tempus Fugit/Interview is a computational fluid dynamics visualization application for which processing is distributed between high performance graphics workstations and supercomputers. Tempus Fugit interactively creates images animated over time from large data sets representing unsteady fluid flow. The companion program Interview provides facilities for the images to be viewed from a second workstation, creating a cooperative visualization environment. The way in which the computation is partitioned between the supercomputer and the workstations is critical to the capability of the application to present simultaneous, identical, animated images of fluid dynamics to more than one user.

1. Introduction

Scientific visualization has become increasingly important in the analysis of large data sets. The pattern-recognition capabilities of the visual sense are utilized to analyze much greater quantities data than is possible with purely numeric approaches [10]. In the study of computational fluid dynamics (CFD) visualization is used to depict the physical characteristics of computationally simulated fluid flow [21].

Despite advances in the delivery of computational power to users of high-performance graphics workstations, there remain visualization applications for which the computational requirements can only be met by supercomputers. Distributed processing is used to provide the user with the combined capabilities of a high performance graphics workstation and a supercomputer under the control of a single application [7, 15, 25]. The supercomputer's capabilities of great processing power, large memory, large disk storage capacity, and fast disk access are necessary to store, access, and calculate over the large data sets endemic to unsteady flow analysis. The high speed graphics of the workstation transform numerically

calculated data into high resolution animated images of flow features. The human-machine interface environment, also provided by the workstation, adds mechanisms for controlling the visualization system.

The greatest impediment to developing systems for visualization of unsteady fluid flow is the size of the data, which constrains interactive response time. Effective use of the large volume, fast access disks on the supercomputer is essential to the development of a system capable of interactively presenting animated images extracted from large time-dependent data sets.

In order to provide an interactive interface for visualization of unsteady fluid flow, the computational tasks must be partitioned between the graphics workstation and the supercomputer in a way which efficiently utilizes the strengths of each environment.

As the visualization application utilizes the advanced features of high-performance graphics workstations and the computational power available through distributed processing with supercomputers, the individual scientist is presented with more and more information-laden images. The ability of the scientist to share the information acquired during the visual analysis process with another scientist becomes more difficult. Images created by such visualization applications are not transportable beyond the workstation without some loss of informational content.

Recently developed computer tools which facilitate collaboration have shown that a WYSIWIS (what you see is what I see) interface is valuable. This type of interface provides for the "presentation of consistent images of shared information to all participants" [31, 32]. Such an interface to a visualization application would allow scientists to see and interact with each other's work through their workstations. Visualization, then, becomes a communication medium and the graphics workstation a platform for the exchange of ideas.

Cooperative visualization of unsteady fluid flow requires the integration of the two main services that computer networks provide, the movement of information from one machine to another and the accessibility of computational resources by one machine from another. How efficiently the integration can be accomplished depends upon how the computation is partitioned between the constituent processors.

Tempus Fugit ("time flies") is a tool for visualizing unsteady fluid flow. Processing in *Tempus Fugit* is distributed between a high performance graphics workstation and a supercomputer. The companion program *Interview* provides facilities to share the supercomputer computational environment with a second workstation, creating a cooperative visualization environment [15].

The development of *Tempus Fugit/Interview* is necessarily an interdisciplinary endeavor involving CFD, graphics, distributed processing, supercomputing, and computer-supported cooperative work (CSCW). The first part of the paper will

describe Tempus Fugit and the requirements imposed by visualization of large time-dependent data sets in a distributed environment of supercomputers and graphics workstations. The second part of the paper will concentrate on Interview and the requirements for sharing the distributed environment created by Tempus Fugit.

2. Background

For CFD, the increased capabilities of supercomputers have been used to dramatically increase the size and complexity of numerical simulations of fluid flow [25]. As the size of the simulations increase, the size of the solution data also increases and can result in immense data sets representing the physical characteristics of a flow field. The flow solution can be steady state, in which the physical properties at each node do not change over time, or unsteady, in which changes in the physical properties are observed over time.

A number of systems are available for visualization of steady state fluid flow [2, 21, 34]. These systems provide the capability of applying a variety of visualization techniques such as particle paths, isoscalar surfaces, cutting planes, etc., on steady state flow solutions to produce static images. Interactive manipulation of viewing position, hidden surface removal, and shading by the application of a lighting model provide important three-dimensional cues in the two-dimensional screen images produced by these systems.

The greatest impediment to developing systems for the visualization of unsteady fluid flow is the size of the data, which constrains interactive response time. Several systems have been developed to analyze time-dependent data sets which will fit in main memory [4, 19, 30]. The greatest difficulty in providing interactive response for unsteady flow visualization emerges when the size of the data exceeds the amount of data which can be contained in main memory.

Smith, *et al.* [30], discuss several methods of reducing unsteady flow solution data to fit into main memory. Each method allows the portrayal of animations of unsteady flow phenomena, but requires a decrease in resolution in time and/or space or a limitation to a single visualization technique. As such, the exploratory nature of the visualization process is inhibited.

Visualization of unsteady fluid flow in a virtual environment has utilized data resident in the main memory of a high-performance graphics workstation [4]. A recent addition to this system distributes processing to a supercomputer. Plans to provide for interactive response for calculation over disk-resident data have been developed [5].

Visual3, a visualization tool for unsteady, unstructured data sets, does not require main memory residence for the data set [18]. However, interactive performance decreases as the data set reaches a size that cannot be contained in main memory.

3. The Computational Environment

One of the main objectives of the Numerical Aerodynamic Simulation (NAS) Program at the NASA Ames Research Center is the provision of a comprehensive computing environment to facilitate computational aerodynamics and fluid dynamics research [1]. To this end, the NAS Processing System Network (NPSN) was developed. The NPSN contains a wide range of computer systems, including several supercomputers (Convex C3240, Cray 2 4/256, Cray YMP 8/256, Intel IPSC 860, TMC 32K CM2) and a small army of Silicon Graphics IRIS workstations. Several networks provide connectivity and a basis for network development and research. These networks include Ethernet, UltraNet, and FDDI.

The main vehicle for distributing computation between supercomputers and workstations in Tempus Fugit/Interview is Distributed Library (dlib) [36]. Like many systems which provide for distributed processing, dlib is a high level interface to network services based on the remote procedure call (RPC) model [3, 12, 33, 35]. However, unlike most of these systems, dlib was developed to provide a service which allows for a conversation of arbitrary length within a single context between client and server. The dlib server process can allocate memory for data storage and manipulation, as well as store state information which persists from call to call. While RPC protocols are frequently likened to local procedure calls without side effects, dlib more closely resembles the extension of the process environment to include the server process.

Using dlib is much like developing a library of routines, say, an I/O library, on a local system. Application code is linked to routines in an I/O library. The I/O library contains simple routines which give access to the I/O devices controlled by the operating system device drivers, as illustrated in figure 1. The I/O device drivers, in turn, control the somewhat more complicated exchange of data with external devices.

To execute a routine on a remote host, all the information necessary to execute the routine in the remote environment must be transmitted over the network to a remote server process. After the routine executes, results must be transmitted back to the local client process. Dlib provides utilities to automatically create the code which performs the network transactions required to invoke and execute the routine in the remote environment and to exchange information between the client and server processes.

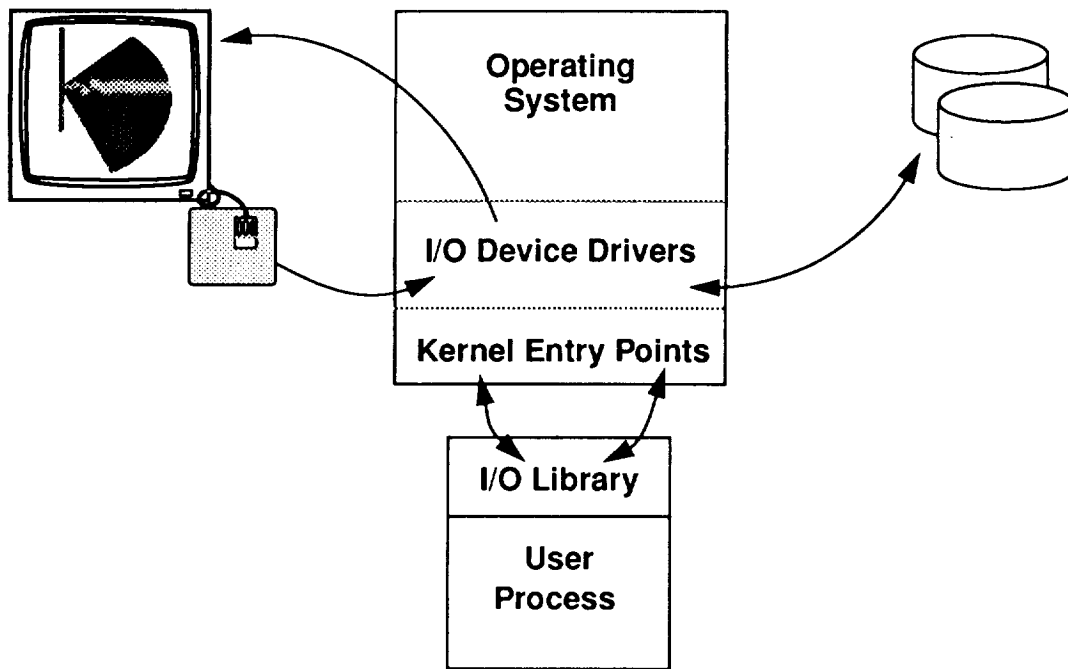


Figure 1: Access to Local I/O Devices Using Local I/O Library.

Due to the persistent nature of the remote environment, dlib is able to coordinate allocation and use of remote memory segments and provide access to remote system utilities. The application, through dlib, can “link” to the remote system’s I/O library, for example, to utilize the remote system’s I/O devices as depicted in figure 2. The illustrated client process can utilize the monitor and mouse via the local I/O library and operating system. The client process can also utilize the remote disk via dlib which communicates with a remote server process. The remote server process has access to the remote disk via the remote I/O library.

Tempus Fugit/Interview uses dlib to allow the workstation clients to exploit the supercomputer’s processing power, large memory, large disk storage capacity, and fast disk access.

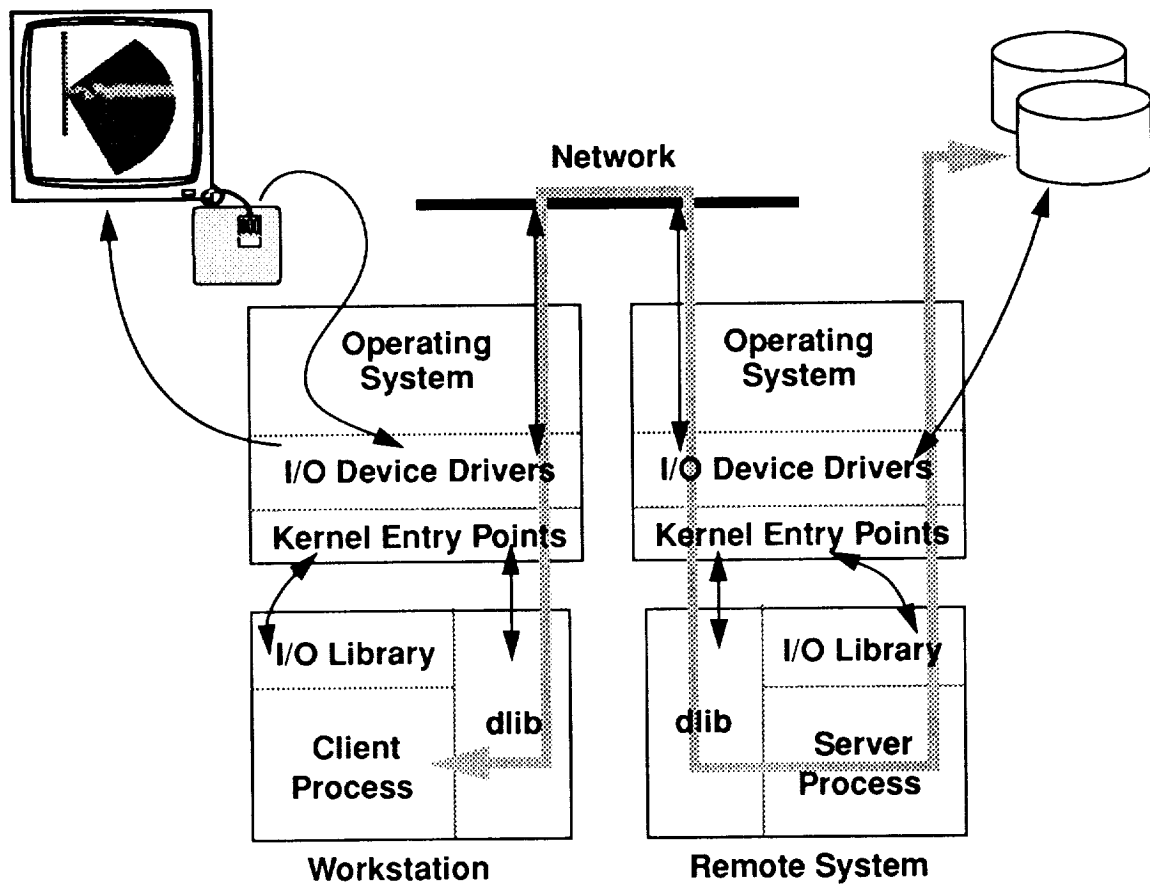


Figure 2: Access to Remote I/O Devices Using Dlib and Remote I/O Library.

4. The CFD Application

CFD research can be divided into three steps: grid generation, numerical simulation, and post-process analysis. A numerical grid is created describing an object and the surrounding space. Flow solvers calculate physical properties of the flow at the nodes of the grid.

A typical data set consists of a grid file, containing the x-, y-, and z-coordinate values of the grid nodes, and a solution file, containing the values for density, energy, and momentum for each grid node. Density and energy are scalar values while momentum is a three-dimensional vector. A steady state solution data set contains a grid file and a solution file. An unsteady solution data set contains a grid file and a solution file per time step. Typical grid sizes can be as large as several million nodes. Due to storage considerations, unsteady solution data sets are usually truncated in some manner but can still consume multiple gigabytes of storage space.

From the density and energy scalar fields and momentum vector field, other scalar

and vector fields can be calculated. The widely used PLOT3D CFD visualization tool developed by Pieter Buning [34] provides facilities for building fields for about one hundred different scalar and vector functions and provides the model for many CFD visualization tools. The FAST CFD visualization tool provides a scalar and vector field calculator as well as a number of built-in functions [2].

A variety of visualization techniques can be applied to these scalar and vector fields for post-process analysis. These include particle paths, isoscalar surfaces, cutting planes, and topology. Graphics techniques are applied to color, shade, light, project and otherwise render images on the workstation monitor.

The process of visualizing CFD can be understood as the transformation of data (figure 3). Raw data is the grid data and solution data produced by flow solvers. The raw data can be processed by applying a function producing a scalar or vector field and further processed and formatted to produce data which can be understood by graphics library routines. This processed data is known as geometry data. Geometry data is rendered to form image data.

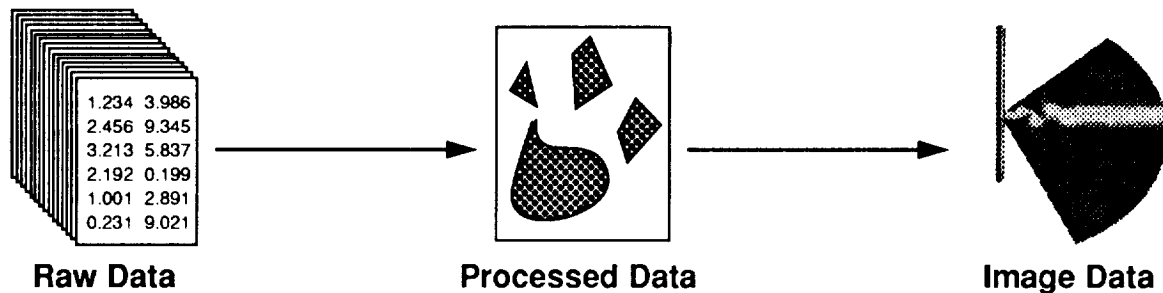


Figure 3: Visualization Process.

Figure 4 shows the relative data sizes for the various steps of the visualization process. The figure for raw data under each visualization technique is the amount of raw data that is used to produce the geometry data. A word size of four bytes is used although for some machines (e.g., Cray 2, Cray YMP) the word size is eight bytes. Differences in the size of raw data processed and the sizes of geometry and image data created are parameters to consider when choosing how to partition computation.

A grid surface is a subset of the curvilinear grid (i -, j -, k -coordinates) for which one of the three components is a constant. The geometry of a grid surface is made up of the x -, y -, z -coordinate locations of the surface and a color value representing the scalar function value at each node for each time step. The image size is calculated for a megapixel display with 24 bits per pixel for color information. Ballpark figures for cutting plane polygons (20,000) and isosurface polygons (50,000) are used. The cutting plane contains a color value for a scalar field. The isosurface includes polygon vertex coordinates and surface normals.

Data Type	Data Description	Size
Grid:	100 x 100 x 100 nodes x 3 coordinates	12 MBytes
Solution:	100 x 100 x 100 nodes x 5 values x 100 time steps	2000 MBytes
Grid Surface:		
Raw:	100 x 100 nodes x 5 values x 100 time steps	20 MBytes
Geometry:	100 x 100 nodes x 1 value x 100 time steps	4 MBytes
Image:	1024 x 1024 pixels x 24 bits x 100 time steps	300 MBytes
Cut Plane:		
Raw:	2 x 100 x 100 x 5 values x 100 time steps	40 MBytes
Geometry:	20,000 polygons x 3 x 3 coordinates x 3 values x 100 time steps	216 MBytes
Image:	1024 x 1024 pixels x 24 bits x 100 time steps	300 MBytes
Isosurface:		
Raw:	100 x 100 x 100 nodes x 5 values x 100 time steps	2000 MBytes
Geometry:	50,000 polygons x 3 x 3 coordinates x 3 x 3 normals x 100 time steps	1620 MBytes
Image:	1024 x 1024 pixels x 24 bits x 100 time steps	300 MBytes

Figure 4: Data Sizes.

5. Partitioning

Distributed processing can be applied to the visualization process by partitioning the computational tasks between a supercomputer and high performance graphics workstations. How this partitioning is accomplished will determine the ability of the visualization system to provide a distributed environment which best utilizes the capabilities of the graphics workstation, the supercomputer, and the network. This section describes several possible partitions: the distributed file system partition, the frame buffer partition, the distributed graphics library partition, and the geometry partition.

Figure 5 illustrates a distributed processing partition in which the raw data resides on the server machine and is made available for processing on the client. The use of distributed file systems such as Network File System (NFS) [27] is often given as an example of distributed processing. This scheme allows the client to utilize the disk resources of the server. Another example of this type of partition is when a server process creates the raw data and delivers it directly to a client process. In CFD visualization this method has been used to preview data as it is produced by a flow solver. Intermediate solutions are produced by a flow solver and transferred over the

network to a visualization application instead of being written to disk.

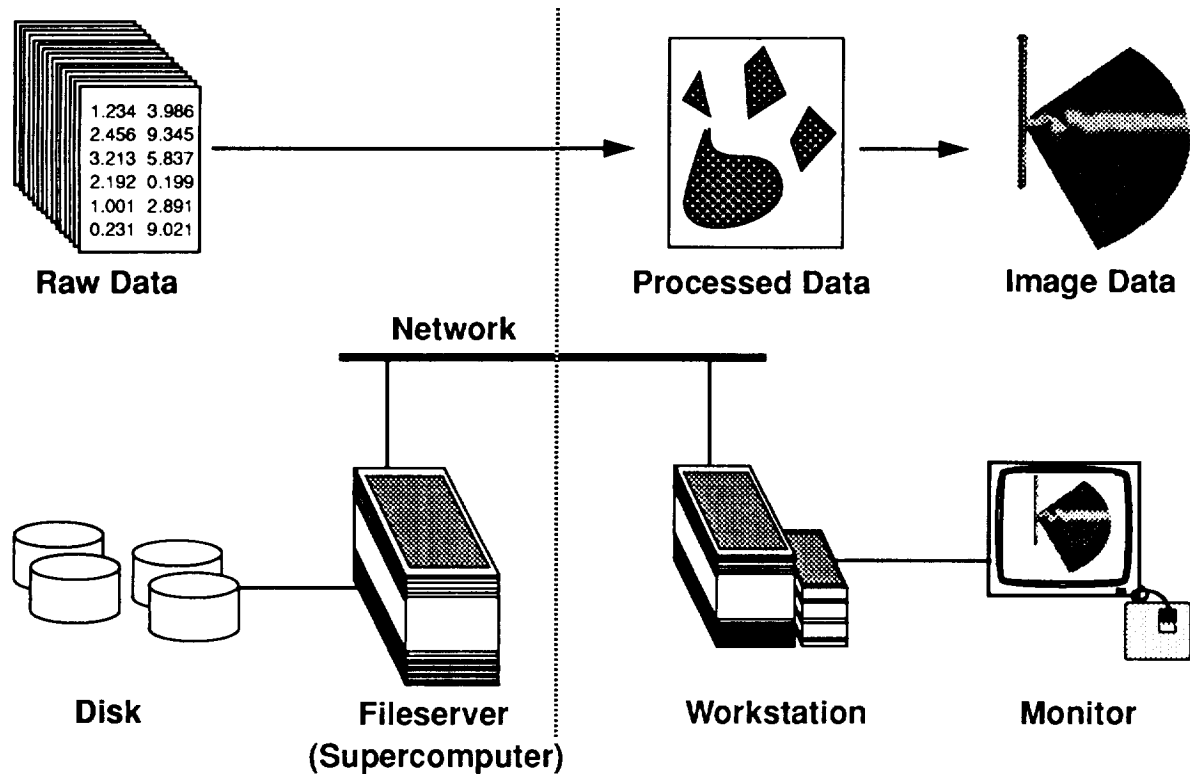


Figure 5: Distributed File System Partition.

A small savings disk access time and space is gained from this method. However, for a visualization application, this partition would only utilize the high disk-to-memory speed of the supercomputer, leaving the computationally intensive processing of the data to the workstation.

The frame buffer partition is illustrated in figure 6. All of the processing is accomplished on one machine and the image is transferred to another for display. The image can be transferred as pixel image data or further transformed into video [14, 17].

This has been found to be useful in that the image display takes place at a different machine and location than the calculation and rendering of the image, avoiding a requirement of geographical proximity of the user to the machine which is carrying out the calculation.

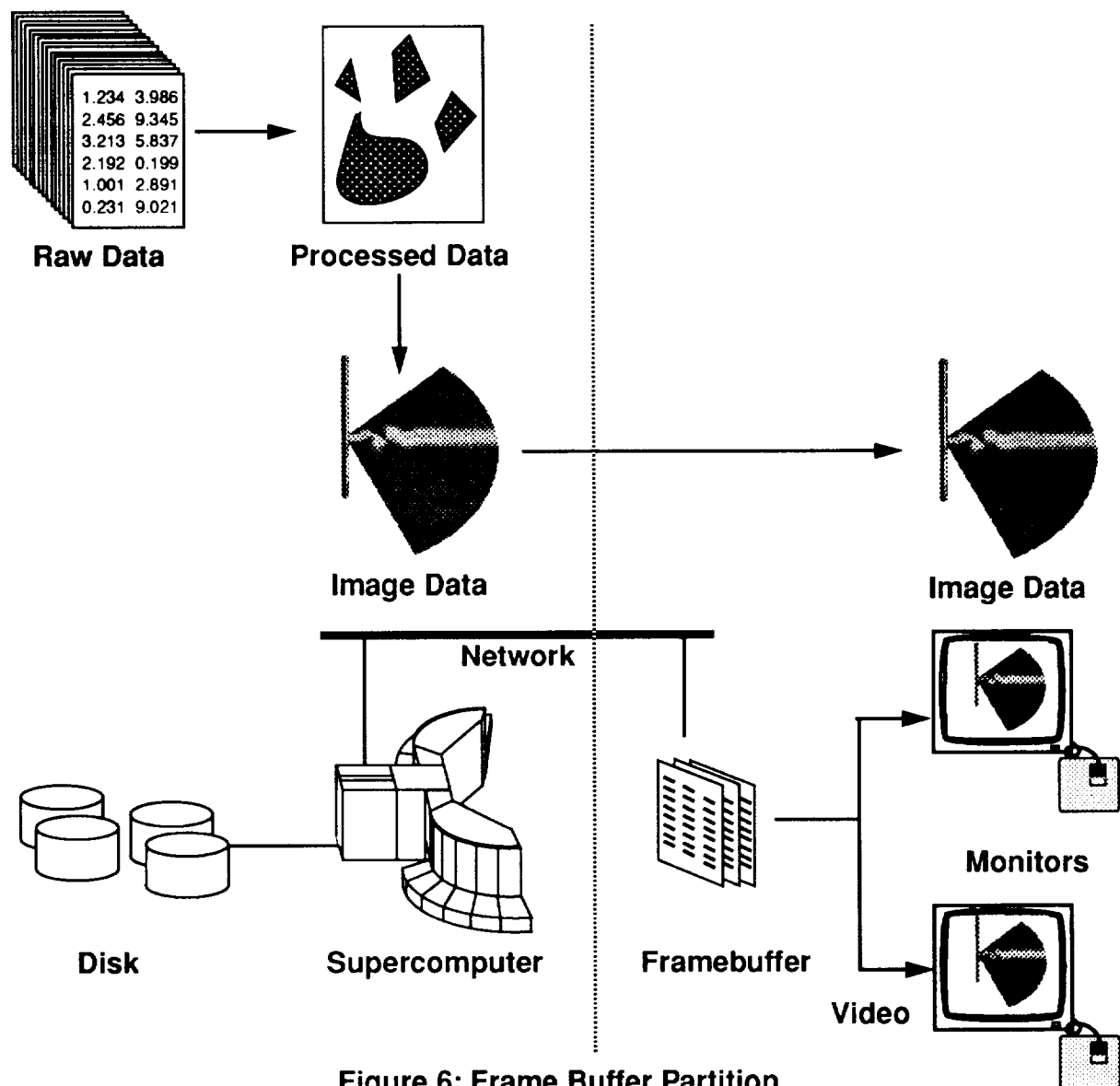


Figure 6: Frame Buffer Partition.

Images can turn out to be quite a large amount of data to transfer over the network (figure 4). The data transfer rate, at 24 frames per second, would be approximately 72 MBytes per second for an image with three MBytes of pixel data. Using NTSC video transmission can decrease the volume of data to be transferred but only at the cost of greatly reducing the quality of the images. The advent of gigabit-per-second networks [6, 8, 13] removes a potential bottleneck in the performance of such distributed applications, but high speed networks are not a panacea. Even though some networks may be capable of transferring data at a rate of one gigabit-per-second, it will be some time before workstations are capable of sending or receiving data at that speed. Even if a one gigabit-per-second throughput rate were attainable by workstations, image data transfer at animation speeds would be difficult to

sustain.

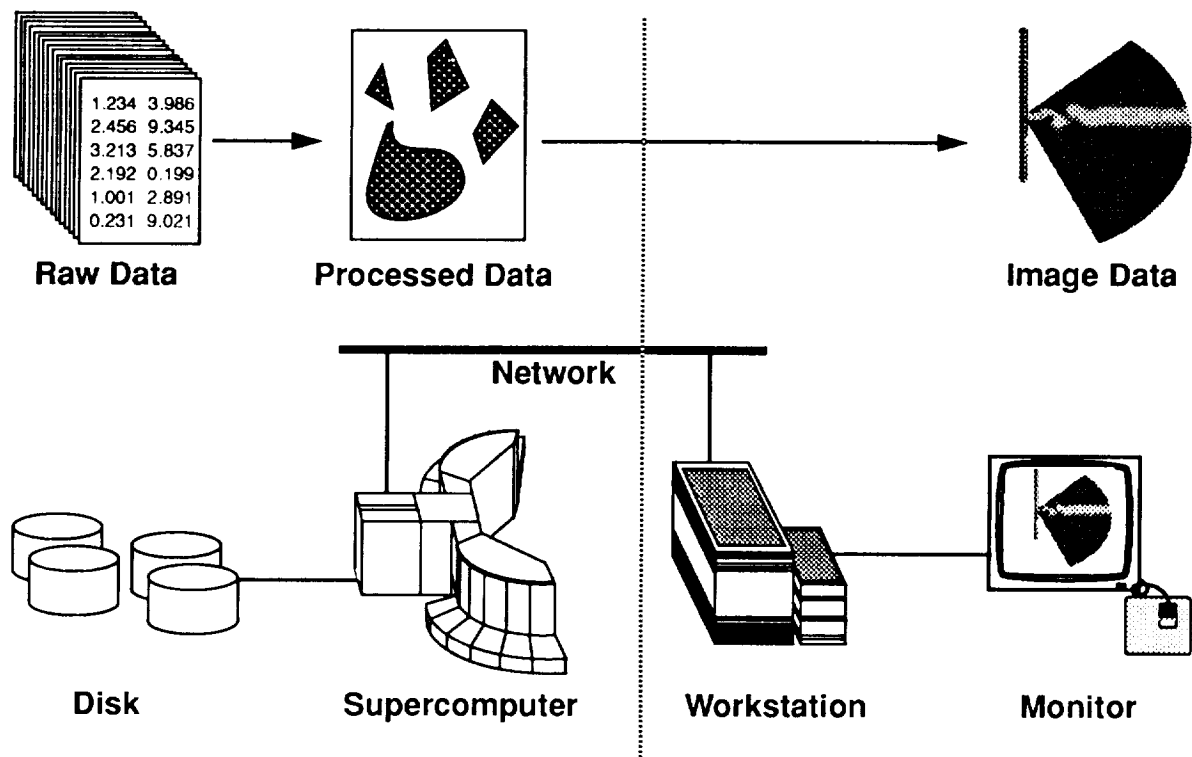


Figure 7: Distributed Graphics Library Partition.

A third partition is at the transition between processed data and the creation of image data (figure 7). Graphics routines transform geometry data into images. The graphics library call sequence may be transferred over the network and executed on another machine. This partitioning has the advantage that the programming effort is carried out on the computation machine (supercomputer) and graphics calls made as though the image creation was local even though it is carried out on another machine. A distributed graphics library performs the network transactions [29]. Network window systems, such as the X window system, are another example of the utilization of the distributed graphics library type of partition [23].

Use of a distributed graphics library partitions the computation in a way which utilizes the strengths of the supercomputer and of the graphics workstation. It is advantageous to be able to use distributed processing which does not involve graphics. This ability is unfortunately not available with the distributed graphics library or network window system approach.

The distributed file system partition, the frame buffer partition, and the distributed graphics library partition are designed to have minimal impact on the basic design of an application which is being modified to distribute processing.

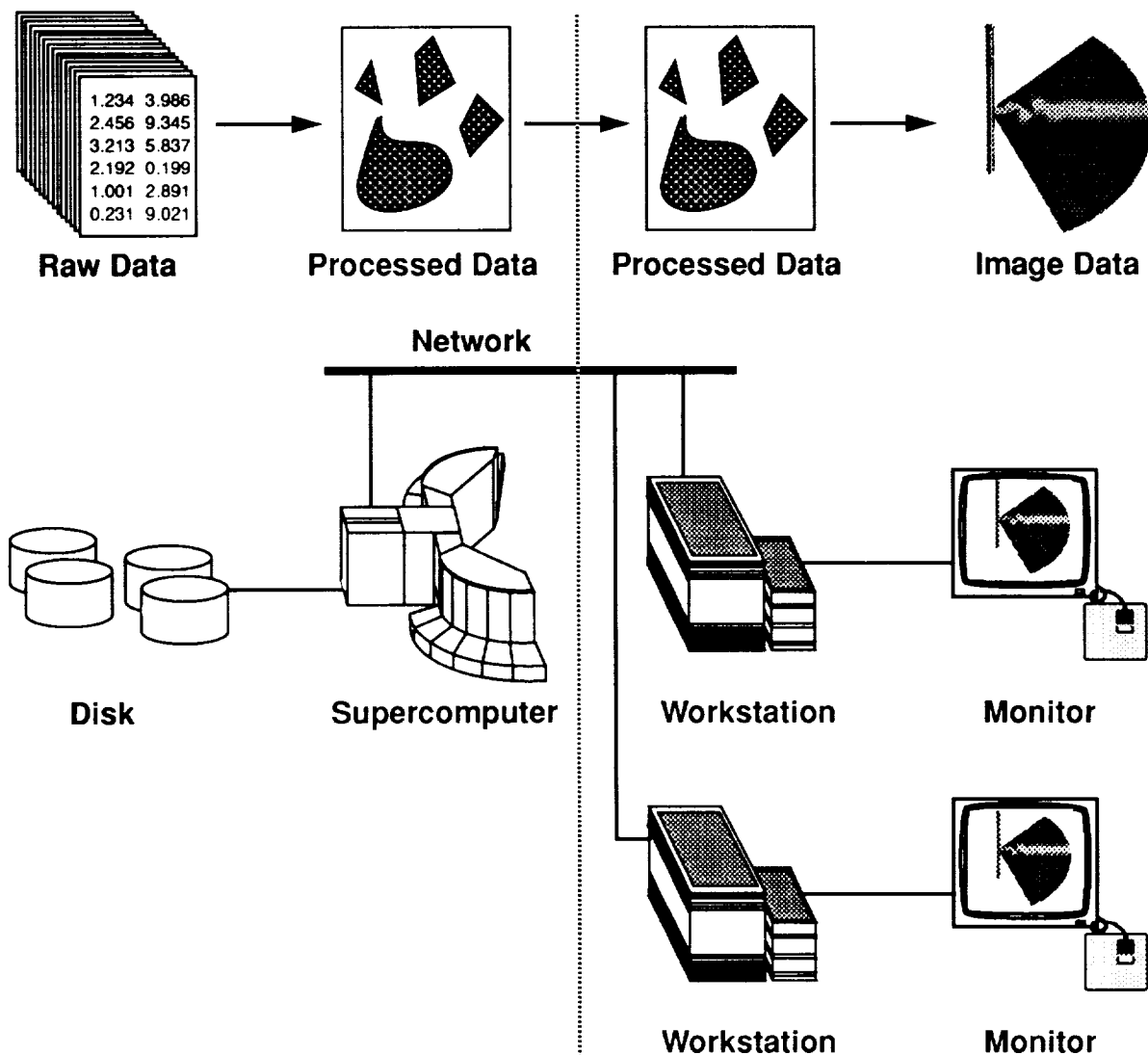


Figure 8: Geometry Partition.

With a distributed graphics library partition each new image requires network transactions. The network transactions and the associated processing overhead can be reduced by moving the geometry data to the workstation. The entire rendering process can then be completed on the workstation without involving the network for transferring additional command information or data (figure 8).

6. Tempus Fugit

Tempus Fugit was designed with the philosophy that with an interactive application a user will be willing to wait for a short period of time, say up to one minute, for a visualization to be presented, if for a great majority of the time the application

provides immediate response. Tempus Fugit provides immediate interactive manipulation of viewing position, which is important for providing three-dimensional cues in the two-dimensional screen images. Animation control is also an important immediate interactive capability, since the main focus of the analysis of unsteady fluid flow is on the variations of the flow over time.

Tempus Fugit is an application for visualizing unsteady fluid flow. The transformation of raw data to geometry data by the application of various visualization techniques is accomplished on the supercomputer and controlled by a mouse-driven interface on the workstation. The geometry data is transferred over a connection to the dlib client on the workstation, hereafter called the Tempus Fugit client, where time-sequenced animated images are produced as in the geometry partition described above (figure 8).

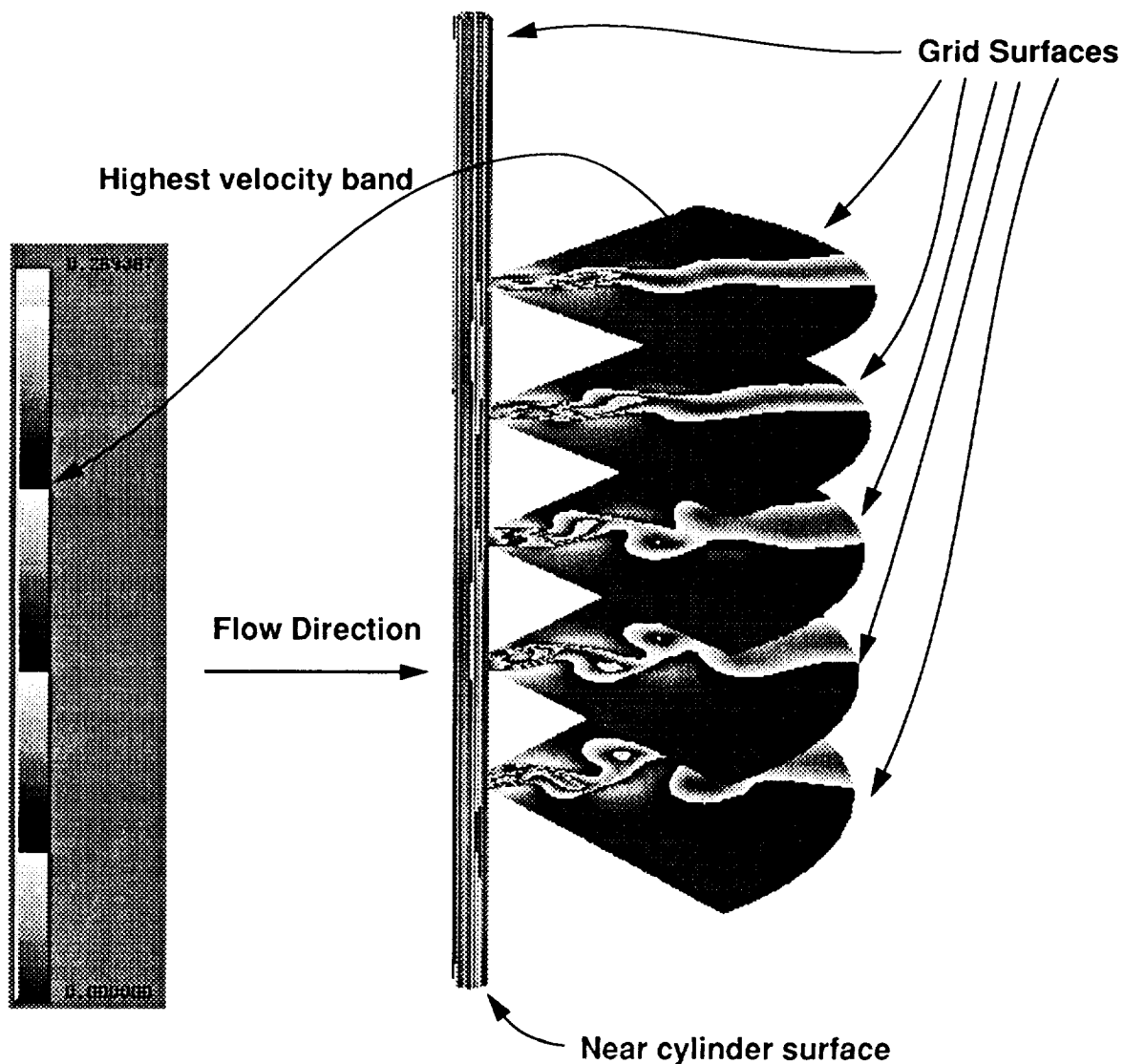


Figure 9: Grid Surfaces Shaded by Velocity Magnitude.

The process of selecting a visualization technique to be applied, transforming raw data to geometry data, transferring the geometry data to the workstation, and rendering the data into an animated image can result, for instance, in an animation of a grid surface. Several grid surfaces can be added to the animation (figure 9).

The process of extracting data from disk, applying a visualization technique to form geometry data, and transferring the geometry over the network, is time consuming. This is the portion of the application which is performed during short wait periods.

The two I/O steps, extracting data from disk and transferring data over the network, are most efficient when applied to large buffers of data. One method of making the entire visualization process more efficient, then, is to maximize the size of disk reads and transfers over the network.

When the size of the raw data exceeds what can be contained in main memory, the organization of the data on disk becomes an important performance factor. Just as one organizes data in main memory to avoid bank conflict for high performance computing applications, the data on disk must be organized for optimal use with visualization applications.

7. Facilitating Collaboration

The high resolution, three-dimensional, color, animated images which are the result of visualizing unsteady fluid flow are not transportable beyond the workstation without some loss of informational content. Much of the three-dimensional character of the images is lost without the capability to perform interactive view transformations. Recording the animated images using video is limited by the resolution of NTSC video encoding and eliminates interactive capability. Color printing or photography is comparable in color resolution and in some cases is superior in image resolution to the workstation graphics monitor. However, much of the information to be analyzed is in the animation of the images and cannot be captured with still images.

The lack of transportability of these images complicates communicating the results of the visualization process to collaborators, and makes the collaboration process that much more difficult. A cooperative visualization tool, which would allow the users to simultaneously view the images as they are created, would create a shared environment for analyzing the images and facilitate the dialog so important to collaboration.

The emerging field of CSCW emphasizes the use of computers to promote collaboration. One model of a collaborative environment is represented in the simple diagram in figure 10. While conversation is serial and ephemeral in nature, shared

space is substantial and provides another medium for communication [28].



Figure 10: Collaborative Environment Model.

Shared access to information makes symbolic representation more concrete. Stefik, *et al.* use the chalkboard as a metaphor for a shared space for information storage [32]. The idea of WYSIWIS follows from this basic model.

With typical visualization applications a single user is “alone” with the data and the images which are the result of the visualization process. When an interesting image is produced on the graphics monitor, it is common in our laboratory to call co-workers to the monitor to see what has been produced. With the shared view of the monitor, the collaborative environment outlined above is created (figure 11). Collaborators who are in another building or another city, however, cannot participate in this interaction.

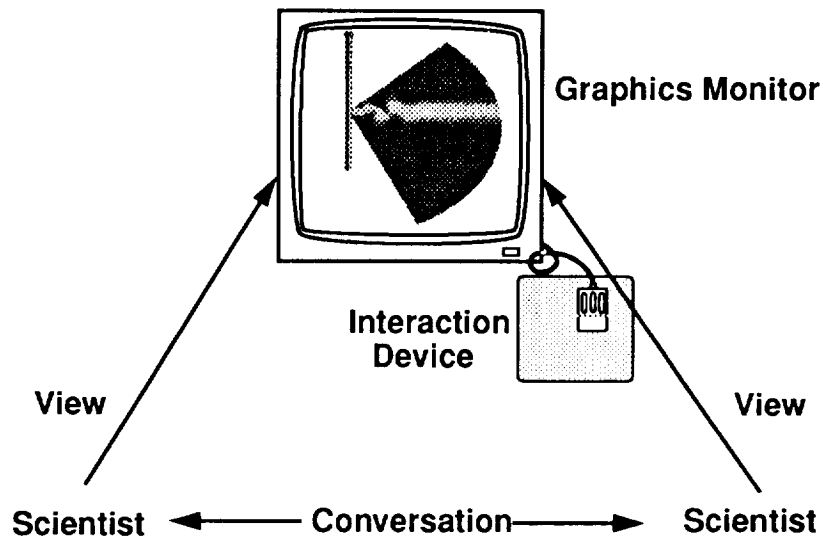


Figure 11: “Gather Around” Collaborative Environment.

Tempus Fugit/Interview builds on the basic model to provide an environment in which distance is not an impediment; the shared space resides on the supercomputer and the images are rendered on separate graphics workstations (figure 12).

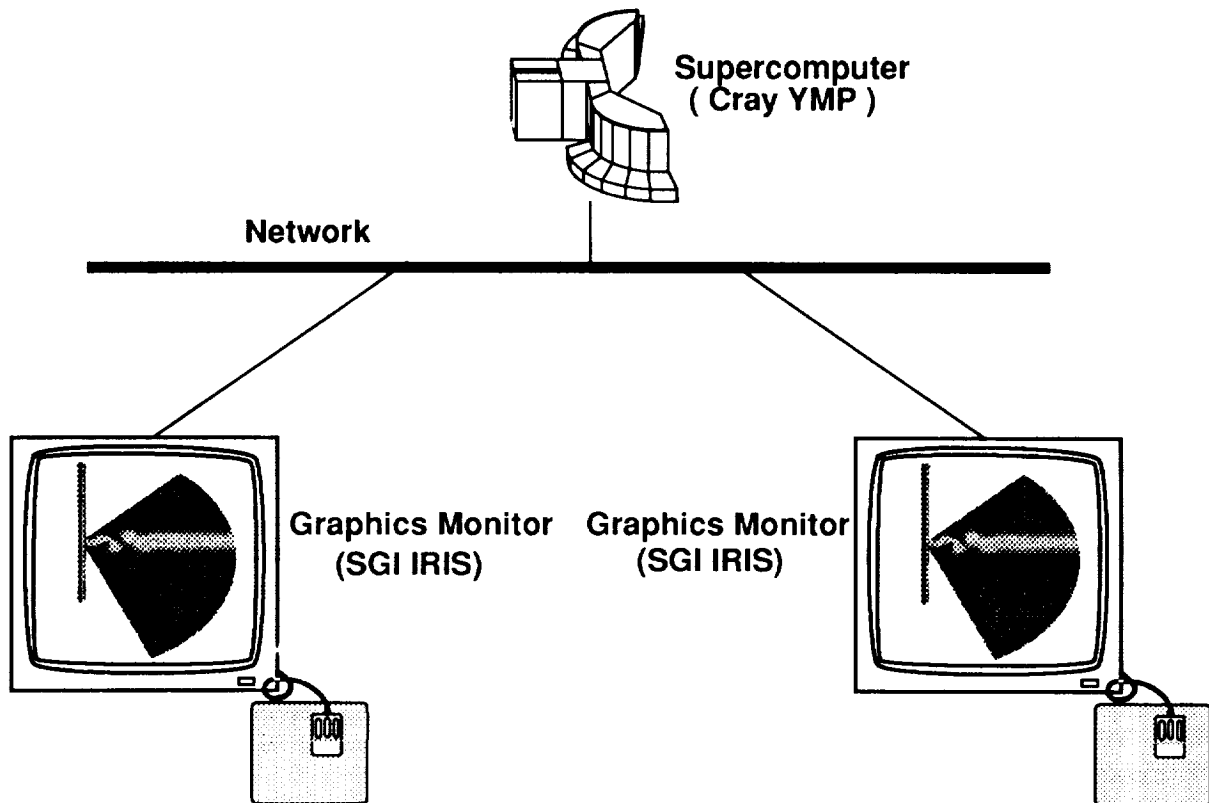


Figure 12: Tempus Fugit/Interview Collaborative Environment.

8. Interview

Dlib was originally designed on a model of one client to one server. To allow multiple clients to share the server process environment, the dlib server was modified to accept more than one connection. Each connection is selected for service by the server process in the sequence that the dlib calls are received. The dlib calls are executed by the server in a single process environment as though there were only one client.

The dlib server executes the transformation of raw data to geometry data on the supercomputer and provides a substrate for the shared space of the collaborative environment model (figure 10).

The Tempus Fugit client will have been active for an arbitrary length of time when Interview is invoked. As such, the image the Tempus Fugit client is presenting may contain a number of graphical objects. Interview creates a connection to the dlib server on the supercomputer and to the Tempus Fugit client (figure 13). The Tempus Fugit client maintains a list of descriptions of the graphical objects it is currently viewing. Upon request this list is sent to the Interview client. From this list, the

Interview client has the information to request the geometry data from the server process using dlib calls in the same way that the Tempus Fugit client requested the data.

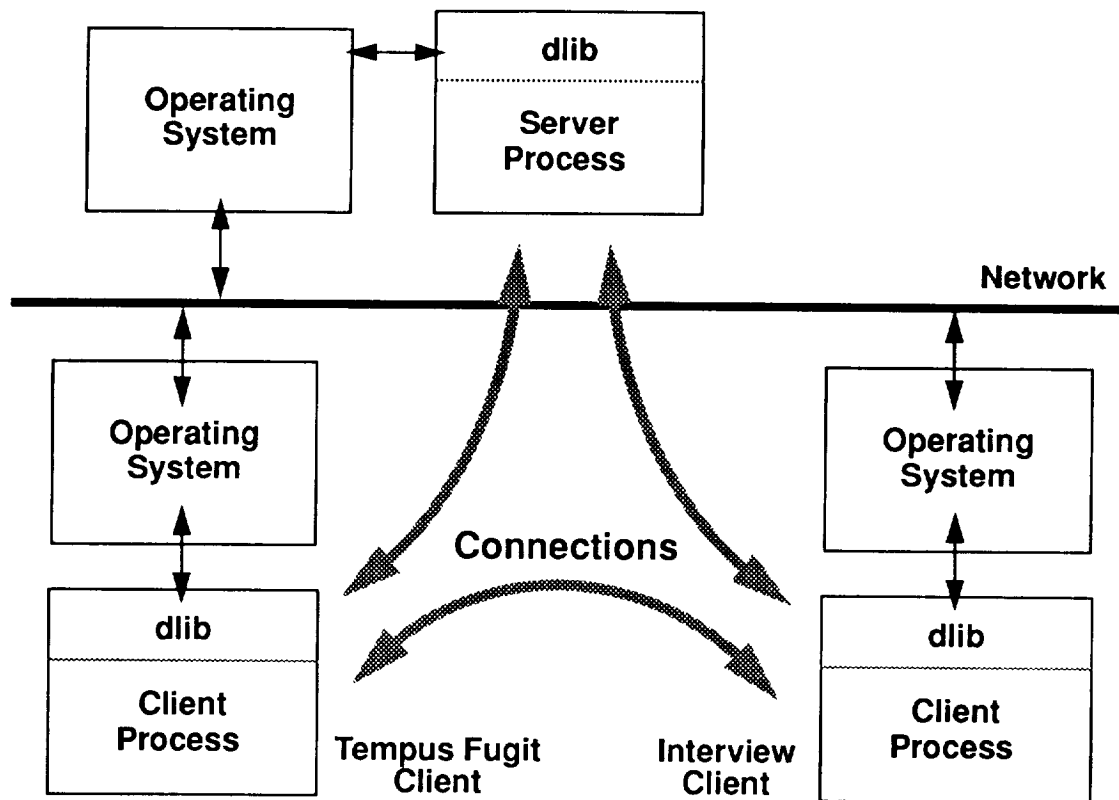


Figure 13: Tempus Fugit / Interview Software Architecture.

When the geometry data is transferred to two workstations, the basis for sharing the information has been established. The application of visualization techniques to the raw data creates the geometry data. With a copy of the geometry data, each workstation can render images distinct from the other workstation. By exchanging the rendering control information the two workstations can generate identical images. So, there are two levels of sharing which can be implemented with a geometry partition, geometry data and rendering controls such as view transformations.

The Interview client is now able to present images created from the same geometry data as the Tempus Fugit client. Interactive controls for view transformations and animation sequencing are individual to each client. Consequently, the two clients at this point are viewing the same three-dimensional geometry data but may have different viewing perspectives.

The ability to individually view the same geometry data is analogous to two people looking at a three-dimensional object, say, an open book. One person can see the title

and author on the front cover, while the other can read the pages. While their views are different, there is a shared context for conversing.

To present an identical image on both monitors simultaneously, rendering controls such as view transformations and animation sequencing must be consistent. The graphics transformation matrix controls the mapping between geometry data and the screen representation. In order for identical images to be viewed by both clients, the rendering control information of one client is sent to the other client. The receiving client loads the transformation matrix into its graphics pipeline, creating an image identical to that of the sending client. The animation sequencing information is used to synchronize the animation frame by frame.

Each client has a set of mouse-driven view transformation controls for rotation, pan, and zoom. To see what is being presented by the other client “tracking mode” is selected. This results in a message sent to the other client to begin sending the rendering control information. The other client continuously updates the rendering control information until tracking mode is deselected and “detached mode” is entered. While in tracking mode, view transformation controls are disabled. While in detached mode, the client is able to control the viewing perspective.

9. Discussion

Tempus Fugit uses the geometry partition to meet the imposing requirements for visualizing complex three-dimensional unsteady fluid flow. The additional requirements of providing a cooperative environment are met by Interview by exploiting the short wait/immediate response paradigm for interactivity and a hybrid of centralized and replicated CSW architectures [15].

With the basic architecture of Tempus Fugit/Interview, optimization methodologies are applied to individual tasks with an effect of improving the efficiency of the overall application. The actual optimization methodologies must balance how long the short wait is against the animation and immediate interaction rates. The profile of where this balance lies is dependent on the visualization technique.

Unfortunately, an optimization for one visualization technique may conflict with an optimization for another visualization technique. For instance, for the current implementation of Tempus Fugit/Interview, sample data is organized to have time dimensions of data arrays ordered sequentially on disk. This optimizes disk reads over spatial subsets for such visualization techniques as grid surfaces, cutting planes, and isosurfaces over subsets of the grid. Integral curve visualization techniques, such as those used by the Virtual Windtunnel [4, 5], perform best with a disk organization with the spatial dimensions of data arrays ordered sequentially. Additional work is required to find an organization for disk-resident data which is effective for both types of visualization techniques.

The geometry partition provides a balance of shared computation by the server process to generate geometry data and replicated computation by the client processes to generate images. This hybrid of the centralized and replicated architectures defined by CSCW researchers is critical to the ability to present to clients both views of a scene from different perspectives and from identical perspectives.

10. Conclusion

The way in which computation is partitioned between constituent processors is an important design consideration for distributed and cooperative applications. Many applications have been implemented for use on a single machine before it becomes desirable to operate the application in a distributed and/or cooperative environment. Processing can be partitioned in ways which require minimal modification to a single processor implementation. However, it is difficult for such partitions to take full advantage of available network facilities. For applications which are designed from the start to be distributed and/or cooperative, efficiencies in the passing of control, in providing shared and private contexts, and in providing computational services should guide how the tasks are partitioned between the constituent machines.

11. Acknowledgments

The author would like to thank E. Lisette Gerald-Yamasaki and Jeff Hultquist for reviewing early versions of this paper and suggesting numerous improvements to the final presentation.

12. References

- [1] Bailey, F. R. Status and projections of the NAS program. In *Computational Mechanics - Advances and Trends*. A. K. Noor, editor New York: American Society of Mechanical Engineers, 1986, 7-21.
- [2] Bancroft, G. V., Merritt, F. J., Plessel, T. C., Kelaita, P. G., McCabe, R. K. and Globus, A. FAST: a multi-processed environment for visualization of computational fluid dynamics. In *Proc. Visualization '90* (San Francisco, CA, Oct. 23-26, 1990) 14-23.
- [3] Birrell, A. D., and Nelson, B. J. Implementing remote procedure calls. *ACM Trans. on Comp. Sys.* 2, 1 (Jan. 1984), 39-59.
- [4] Bryson, S. and Levit, C. The virtual windtunnel: an environment for the exploration of three-dimensional unsteady flows. In *Proc. Visualization '91* (San Diego, CA, Oct. 22-25, 1991), 17-24.

- [5] Bryson, S. and Gerald-Yamasaki, M. J. The distributed virtual windtunnel. (submitted to Supercomputing '92).
- [6] Chlamtac, I. and Franta, W. R. Rationale, directions, and issues surrounding high speed networks. 78, 1 (Jan. 1990), 94-120.
- [7] Choi, D. and Levit, C. Implementation of a distributed graphics system. *Internat. J. Supercomput. Appl.* (Winter 1987), 82-95.
- [8] Clinger, M. Very high speed network prototype development: Measurement of effective transfer rates. In a report to NASA Ames Research Center in satisfaction of Contract #NAS2-12332/CTO#9, (Oct. 1989).
- [9] Crowley, T., Milazzo, P., Baker, E., Forsdick, H. and Tomlinson, R. MMconf: an infrastructure for building shared multimedia applications. In *Proc. CSCW '90* (Los Angeles, CA, Oct. 7-10, 1990) New York, NY: ACM (Order No: 612900), 329-342,
- [10] DeFanti, T. A., Brown, M. D., and McCormick, B. H. Visualization - Expanding scientific and engineering research opportunities. *Computer* 22, 8 (Aug. 1989), 12-25.
- [11] Dewan, P. and Choudhary, R. Flexible user interface coupling in a collaborative system. In *Proc. CHI '91* (New Orleans, LA, Apr. 27-May 2, 1991) New York, NY: ACM (Order No: 608910), 41-48.
- [12] Dineen, T. H., Leach, P. J., Mishkin, N. W., Pato, J. N., and Wyatt, G. L. The network computing architecture and system: an environment for developing distributed applications. *Proceedings of Summer Usenix* (June 1987), 385-398.
- [13] Farber, D. Gigabit network testbeds. *Computer* 23, 9 (Sep. 1990), 77-79.
- [14] Fowler, Jr., J. D., and McGowen, M. Design and implementation of a supercomputer frame buffer system. In *Proc. Supercomputing '88* (Orlando, Florida, Nov. 14-18, 1988) Washington, DC: IEEE Computer Society Press (Order No. 882), 140-147.
- [15] Gerald-Yamasaki, M. Cooperative visualization of computational fluid dynamics. *NAS Applied Research Technical Report RNR-92-007* (submitted to CSCW '92).
- [16] Grudin, J. CSCW: the convergence of two development contexts. In *Proc. CHI '91* (New Orleans, LA, Apr. 27-May 2, 1991) New York, NY: ACM (Order No: 608910), 91-97.
- [17] Haber, R. B., and McNabb, D. A. Eliminating distance in scientific computing:

- an experiment in televisualization. *Internat. J. Supercomput. Appl.* 4, 4, (Winter, 1990), 71-89.
- [18] Haimes, R. and Giles, M. Visual3: interactive unsteady unstructured 3d visualization. American Institute of Aeronautics and Astronautics (AIAA) paper 91-0794. AIAA 29th Aerospace Sciences Meeting (Reno, NV, Jan. 7-10, 1991).
 - [19] Hibbard, B. and Santek, D. The vis-5d system for easy interactive visualization. In *Proc. Visualization '90* (San Francisco, CA, Oct. 23-26, 1990) 28-35.
 - [20] Jespersen, D. and Levit, C. Numerical simulation of flow past a tapered cylinder. AIAA paper 91-0751. AIAA 29th Aerospace Sciences Meeting. (Reno, Nevada, Jan. 7-10, 1991).
 - [21] Lasinski, T., Buning, P., Choi, D., Rogers, S., Bancroft, G. and Merritt, F. Flow visualization of CFD using graphics workstations. *Proc. AIAA 8th Computational Fluid Dynamics Conf.* (Honolulu, Hawaii, June 9-11, 1987) AIAA Paper 87-1180, 814-820.
 - [22] Lauwers, J. C. and Lantz, K. A. Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems. In *Proc. CHI '90* (Seattle, WA, Apr. 1-5, 1990) New York, NY: ACM (Order No: 608900), 303-311.
 - [23] O'Reilly & Associates Inc. *X Window System Series* Sebastapol: O'Reilly, 1990.
 - [24] Patterson, J. F., Hill, R. D., Rohall, S. L. and Meeks, W. S. Rendezvous: an architecture for synchronous multi-user applications. In *Proc. CSCW '90* (Los Angeles, CA, Oct. 7-10, 1990) New York, NY: ACM (Order No: 612900), 317-328.
 - [25] Rogers, S. E., Buning, P. G., and Merrit, F. J. Distributed interactive graphics applications in computational fluid dynamics. *Internat. J. Supercomput. Appl.* 1, 4 (Winter 1987), 96-105.
 - [26] Salzman, D. Visualization in scientific computing: Summary of an NSF-sponsored panel report on graphics, image processing, and workstations. *Internat. J. Supercomput. Appl.* 1, 4 (Winter 1987), 106-108.
 - [27] Sandberg, R. The sun network file system: design, implementation, and experience. Sun Microsystems, Inc. (1986).
 - [28] Schrage, M. *Shared Minds*. New York: Random House, 1990

- [29] Silicon Graphics Computer Systems. Using the distributed graphics library. *4-Sight Programmer's Guide* Document Number 007-2001-030 (1990).
- [30] Smith, M. H., Van Dalsem, W. R., Dougherty, F. C., and Buning, P. G. Analysis and visualization of complex unsteady three-dimensional flows. AIAA Paper 89-0139. AIAA 27th Aerospace Sciences Meeting (Reno, NV, Jan. 9-12, 1989).
- [31] Stefik, M., Bowbrow, D. G., Foster, G., Lanning, S. and Tatar, D. WYWIWIS revised: early experiences with multiuser interfaces. *ACM Trans. on Office Info. Sys.* 5, 2 (Apr 1987), 147-167.
- [32] Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Commun. ACM* 30, 1 (Jan. 1987), 32-47.
- [33] Sun Microsystems. *Request for Comment #1057* Network Working Group (June, 1988).
- [34] Walatka, P. P. and Buning, P. G. Plot3d user's manual. NASA Technical Memorandum 101067, NASA Ames Research Center.
- [35] Xerox Corporation. Courier: the remote procedure call protocol. *Xerox System Integration Standard (XSYS) 038112*, (Dec. 1981).
- [36] Yamasaki, M. Distributed library. *NAS Applied Research Technical Report RNR-90-008* (Apr. 1990).

Following Page:

This image is representative of the color and resolution of the images which can be produced with Tempus Fugit/Interview. The display contains a collection of grid surfaces shaded by cross flow velocity and isosurfaces for several values of cross flow velocity over two subranges of the data.

Second Following Page:

This image shows both Tempus Fugit and Interview clients invoked on the same workstation. The Tempus Fugit window is drawn with a white background and the Interview window is drawn with a black background. Each client has a view transformation control window showing.

The same set of grid surfaces is displayed in the Tempus Fugit window as in the Interview window, but from different viewing perspectives (top image)

In bottom image the Tempus Fugit client has selected tracking mode. This results in the Interview client sending its rendering control information to the Tempus Fugit client. Timing information in the rendering control information is used to synchronize the animations. Changes in viewing perspective as controlled by the Interview client are transmitted to the tracking client, Tempus Fugit, to maintain the identical images through application of pan, zoom and rotation controls.

